CONTEXT FREE DOCUMENT PORTIONS

Copyright Notice

Field of the Invention

The present invention relates generally to use of the Extensible Markup Language (XML). More particularly, the present invention relates to inserting a portion of XML data into a document outside the context of a document from which the portion is obtained.

Background of the Invention

Computer software applications allow users to create a variety of documents to assist them in work, education, and leisure. For example, popular word processing applications allow users to create letters, articles, books, memoranda, and the like. Spreadsheet applications allow users to store, manipulate, print, and display a variety of alphanumeric data. Such applications have a number of well-known strengths including rich editing, formatting, printing, calculation, and on-line and off-line editing.

Methods and systems have been developed for representing entire documents and all associated document properties and objects as an Extensible Markup

Language (XML) representation. There are many components that make up such a document such as paragraphs, tables, styles, fonts, lists, and the like. Some components of a document reference other components for providing a first component with structural limitations. For example, a paragraph in a document might reference a particular style setting that would define how the paragraph is to appear in a document. For another example, a paragraph component of a document may be part of a particular list structure in a document. Because there are a number of properties and objects that other objects of a document may reference, such as styles, there is typically a need in a document for a header in which various properties and objects, such as style and font definitions, are located. Accordingly, when a paragraph, for example, references a particular style setting, an application parsing the document will know where to look to find a definition of the style setting referenced by the paragraph.

Unfortunately, a problem occurs when a user attempts to add additional content to a particular document where the additional content contains or references objects and properties such as styles and fonts apart from the objects and properties of the document to which the content is being added. For example, if a user desires to add a couple of paragraphs to an XML-formatted document, where the added paragraphs reference a particular style, it is necessary to determine if the particular style already exists in the document to which the paragraphs are added. If the particular style does exist in the document, it is then necessary to ensure that the style existing in the document contains the same properties as are expected for the added paragraphs. If the style does not exist, it is necessary to insert new style information for the added paragraphs into the header of the document. Either way, rather than simply dealing with the content (paragraphs) being added to the document, the user must deal with the entire document.

It is with respect to these and other considerations that the present invention has been made.

Summary of the Invention

The present invention solves the above and other problems by providing self-describing portions of text or data in a document. According to an aspect of the invention, portions of a document, such as individual paragraphs or groups of paragraphs or fragments of text, are provided self-describing properties such as style, font, list type, and the like. If such a portion of a document is subsequently copied or moved to a second document or to a different location in the first document, the self-describing properties provided for the portion travel with the portion to the second document or to the different location in the first document. Consequently, an application preparing and displaying the second document or the first document may consume and display the portion according to the properties provided for the portion. Apart from copying or moving text from one document to another, the self-describing properties of the text portion allow users and software applications who desire to or who are required to parse and manipulate a file to apply rich editing to portions of the file or document without having to parse the entire document to manage properties associated with the entire document or file.

According to another aspect of the invention, an Extensible Markup Language (XML) schema is provided for providing an XML representation of specified portions of an XML-formatted document such that each specified portion carries with it its own properties such as styles, list types, fonts, and the like. If one of such specified portions of the document is copied or moved to a different document or is , a consuming application operating the different document need only parse the imported text or data portion to determine specific properties associated with the imported text, such as styles, list types, fonts, etc. Accordingly, the consuming application need not parse the header information for the entire document into which the specified portion is inserted, nor does the consuming application have to determine whether any properties associated with the imported portion conflict with definitions associated with similar or like properties applied to the entire document.

These and other features, advantages, and aspects of the present invention may be more clearly understood and appreciated from a review of the

following detailed description of the disclosed embodiments and by reference to the appended drawings and claims.

Brief Description of the Drawings

Fig. 1 is a simplified block diagram of a computing system and associated peripherals and network devices that provide an exemplary operating environment for the present invention.

Fig. 2 is a simplified block diagram illustrating interaction between software objects according to an object-oriented programming model.

Fig. 3 is a block diagram illustrating interaction between a document, an attached schema file, and a schema validation functionality model.

Fig. 4 is a block diagram illustrating interaction between a first document and a second document where a portion of text is copied or moved from the second document to the first document according to embodiments of the present invention.

Fig. 5 is a block diagram illustrating interaction between a first document and a second document where a portion of text is copied or moved from the second document to the first document according to embodiments of the present invention.

Detailed Description of the Preferred Embodiment

Embodiments of the present invention are directed to methods and systems for inserting a portion of text or data into a document where the inserted portion of text or data carries with it its own specific formatting properties such as specified styles, fonts, list types, and the like. In the following detailed description, references are made to the accompanying drawings that forma part hereof, and in which are shown by way of illustrations specific embodiments or examples. These embodiments may be combined, other embodiments may be utilized, and structural changes may be made without departing from the spirit or scope of the present

invention. The following detailed description is therefore not to be taken in a limiting senses and the scope of the present invention is defined by the appended claims and their equivalents.

Referring now to the drawings, in which like numerals represent like elements through the several figures, aspects of the present invention and the exemplary operating environment will be described. Fig. 1 and the following discussion are intended to provide a brief, general description of a suitable computing environment in which the invention may be implemented. While the invention will be described in the general context of program modules that execute in conjunction with an application program that runs on an operating system on a personal computer, those skilled in the art will recognize that the invention may also be implemented in combination with other program modules.

Generally, program modules include routines, programs, components, data structures, and other types of structures that perform particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations, including hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, minicomputers, mainframe computers, and the like. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

Turning now to Fig. 1, illustrative computer architecture for a personal computer 2 for practicing the various embodiments of the invention will be described. The computer architecture shown in Fig. 1 illustrates a conventional personal computer, including a central processing unit 4 ("CPU"), a system memory 6, including a random access memory 8 ("RAM") and a read-only memory ("ROM") 10, and a system bus 12 that couples the memory to the CPU 4. A basic input/output system containing the basic routines that help to transfer information between elements within the computer, such as during startup, is stored in the ROM 10. The personal computer 2 further

includes a mass storage device 14 for storing an operating system 16, application programs, such as the application program 305, and data.

The mass storage device 14 is connected to the CPU 4 through a mass storage controller (not shown) connected to the bus 12. The mass storage device 14 and its associated computer-readable media, provide non-volatile storage for the personal computer 2. Although the description of computer-readable media contained herein refers to a mass storage device, such as a hard disk or CD-ROM drive, it should be appreciated by those skilled in the art that computer-readable media can be any available media that can be accessed by the personal computer 2.

By way of example, and not limitation, computer-readable media may comprise computer storage media and communication media. Computer storage media includes volatile and non-volatile, removable and non-removable media implemented in any method or technology for storage of information such as computer-readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EPROM, EEPROM, flash memory or other solid state memory technology, CD-ROM, DVD, or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by the computer.

According to various embodiments of the invention, the personal computer 2 may operate in a networked environment using logical connections to remote computers through a TCP/IP network 18, such as the Internet. The personal computer 2 may connect to the TCP/IP network 18 through a network interface unit 20 connected to the bus 12. It should be appreciated that the network interface unit 20 may also be utilized to connect to other types of networks and remote computer systems. The personal computer 2 may also include an input/output controller 22 for receiving and processing input from a number of devices, including a keyboard or mouse (not shown). Similarly, an input/output controller 22 may provide output to a display screen, a printer, or other type of output device.

As mentioned briefly above, a number of program modules and data files may be stored in the mass storage device 14 and RAM 8 of the personal computer 2, including an operating system 16 suitable for controlling the operation of a networked personal computer, such as the WINDOWS XP operating system from MICROSOFT CORPORATION of Redmond, Washington. The mass storage device 14 and RAM 8 may also store one or more application programs. In particular, the mass storage device 14 and RAM 8 may store an application program 305 for creating and editing an electronic document 310. For instance, the application program 305 may comprise a word processing application program, a spreadsheet application, a contact application, and the like. Application programs for creating and editing other types of electronic documents may also be used with the various embodiments of the present invention. A schema file 330 and a namespace/schema library 400, described below, are also shown.

Exemplary embodiments of the present invention are implemented by communications between different software objects in an object-oriented programming environment. For purposes of the following description of embodiments of the present invention, it is useful to briefly to describe components of an object-oriented programming environment. Fig. 2 is a simplified block diagram illustrating interaction between software objects according to an object-oriented programming model. According to an object-oriented programming environment, a first object 210 may include software code, executable methods, properties, and parameters. Similarly, a second object 220 may also include software code, executable methods, properties, and parameters.

A first object 210 may communicate with a second object 220 to obtain information or functionality from the second object 220 by calling the second object 220 via a message call 230. As is well know to those skilled in the art of object-oriented programming environment, the first object 210 may communicate with the second object 220 via application programming interfaces (API) that allow two disparate software objects 210, 220 to communicate with each other in order to obtain information and functionality from each other. For example, if the first object 210 requires the functionality provided by a method contained in the second object 220, the

first object 210 may pass a message call 230 to the second object 220 in which the first object identifies the required method and in which the first object passes any required parameters to the second object required by the second object for operating the identified method. Once the second object 220 receives the call from the first object, the second object executes the called method based on the provided parameters and sends a return message 250 containing a value obtained from the executed method back to the first object 210.

For example, in terms of embodiments of the present invention, and as will be described below, a first object 210 may be a third party customized application that passes a message to a second object such as an Extensible Markup Language schema validation object whereby the first object identifies a method requiring the validation of a specified XML element in a document where the specified XML element is a parameter passed by the first object with the identified method. Upon receipt of the call from the first object, according to this example, the schema validation object executes the identified method on the specified XML element and returns a message to the first object in the form of a result or value associated with the validated XML element. Operation of object-oriented programming environments, as briefly described above, are well known to those skilled in the art.

As described below, embodiments of the present invention are implemented through the interaction of software objects in the use, customization, and application of components of the Extensible Markup Language (XML). Fig. 3 is a block diagram illustrating interaction between a document, an attached schema file, and a schema validation functionality module. As is well known to those skilled in the art, the Extensible Markup Language (XML) provides a method of describing text and data in a document by allowing a user to create tag names that are applied to text or data in a document that in turn define the text or data to which associated tags are applied. For example referring to Fig. 3, the document 310 created with the application 305 contains text that has been marked up with XML tags 315, 320, 325. For example, the text "Greetings" is annotated with the XML tag <title>. The text "My name is Sarah" is annotated with the <body> tag. According to XML, the creator of the <title> and

<body> tags is free to create her own tags for describing the tags to which those tags will be applied. Then, so long as any downstream consuming application or computing machine is provided instructions as to the definition of the tags applied to the text, that application or computing machine may utilize the data in accordance with the tags. For example, if a downstream application has been programmed to extract text defined as titles of articles or publications processed by that application, the application may parse the document 310 and extract the text "Greetings," as illustrated in Fig. 3 because that text is annotated with the tag <title>. The creator of the particular XML tag naming for the document 310, illustrated in Fig. 3, provides useful description for text or data contained in the document 310 that may be utilized by third parties so long as those third parties are provided with the definitions associated with tags applied to the text or data.

According to embodiments of the present invention, the text and XML markup entered into the document 310 may be saved according to a variety of different file formats and according to the native programming language of the application 305 with which the document 310 is created. For example, the text and XML markup may be saved according to a word processing application, a spreadsheet application, and the like. Alternatively, the text and XML markup entered into the document 310 may be saved as an XML format whereby the text or data, any applied XML markup, and any formatting such as font, style, paragraph structure, etc. may be saved as an XML representation. Accordingly, downstream or third party applications capable of understanding data saved as XML may open and consume the text or data thus saved as an XML representation. For a detailed discussion of saving text and XML markup and associated formatting and other attributes of a document 310 as XML, see U.S. Patent Application entitled "Word Processing Document Stored in a Single XML File that may be Manipulated by Applications that Understanding XML," U.S. Serial No. 10/187,060, filed June 28, 2002, which is incorporated herein by reference as if fully set out herein. An exemplary schema in accordance with the present invention is disclosed beginning on page 11 in an application entitled "Mixed Content Flexibility," Serial No.

_____, Docket No. 60001.0275US01, filed December 2, 2003, which is hereby incorporated by reference in its entirety.

In order to provide a definitional framework for XML markup elements (tags) applied to text or data, as illustrated in Fig. 3, XML schema files are created which contain information necessary for allowing users and consumers of marked up and stored data to understand the XML tagging definitions designed by the creator of the document. Each schema file also referred to in the art as a Namespace or XSD file preferably includes a listing of all XML elements (tags) that may be applied to a document according to a given schema file. For example, a schema file 330, illustrated in Fig. 3, may be a schema file containing definitions of certain XML elements that may be applied to a document 310 including attributes of XML elements or limitations and/or rules associated with text or data that may be annotated with XML elements according to the schema file. For example, referring to the schema file 330 illustrated in Fig. 3, the schema file is identified by the Namespace "intro" the schema file includes a root element of <intro card>.

According to the schema file 330, the <intro card> element serves as a root element for the schema file and also as a parent element to two child elements <title> and <body>. As is well known to those skilled in the art, a number of parent elements may be defined under a single root element, and a number of child elements may be defined under each parent element. Typically, however, a given schema file 330 contains only one root element. Referring still to Fig. 3, the schema file 330 also contains attributes 340 and 345 to the <title> and <body> elements, respectfully. The attributes 340 and 345 may provide further definition or rules associated with applying the respective elements to text or data in the document 310. For example, the attribute 345 defines that text annotated with the <title> element must be less than or equal to twenty-five characters in length. Accordingly, if text exceeding twenty-five characters in length is annotated with the <title> element or tag, the attempted annotation of that text will be invalid according to the definitions contained in the schema file 330.

By applying such definitions or rules as attributes to XML elements, the creator of the schema may dictate the structure of data contained in a document

associated with a given schema file. For example, if the creator of a schema file 330 for defining XML markup applied to a resume document desires that the experience section of the resume document contain no more than four present or previous job entries, the creator of the schema file 330 may define an attribute of an <experience> element, for example, to allow that no more than four present or past job entries may be entered between the <experience> tags in order for the experience text to be valid according to the schema file 330. As is well known to those skilled in the art, the schema file 330 may be attached to or otherwise associated with a given document 310 for application of allowable XML markup defined in the attached schema file to the document 310. According to one embodiment, the document 310 marked up with XML elements of the attached or associated schema file 330 may point to the attached or associated schema file by pointing to a uniform resource identifier (URI) associated with a Namespace identifying the attached or associated schema file 330.

According to embodiments of the present invention, a document 310 may have a plurality of attached schema files. That is, a creator of the document 310 may associate or attach more than one schema file 330 to the document 310 in order to provide a framework for the annotation of XML markup from more than one schema file. For example, a document 310 may contain text or data associated with financial data. A creator of the document 310 may wish to associate XML schema files 330 containing XML markup and definitions associated with multiple financial institutions. Accordingly, the creator of the document 310 may associate an XML schema file 330 from one or more financial institutions with the document 310. Likewise, a given XML schema file 330 may be associated with a particular document structure such as a template for placing financial data into a desirable format.

According to embodiments of the present invention, a collection of XML schema files and associated document solutions may be maintained in a Namespace or schema library located separately from the document 310. The document 310 may in turn contain pointers to URIs in the Namespace or schema library associated with the one or more schema files attached to otherwise associated with the document 310. As the document 310 requires information from one or more associated schema files, the

document 310 points to the Namespace or schema library to obtain the required schema definitions. For a detailed description of the use of an operation of Namespace or schema libraries, see U.S. Patent Application entitled "System and Method for Providing Namespace Related Information," U.S. Serial No. 10/184,190, filed June 27, 2002, and U.S. Patent Application entitled "System and Method for Obtaining and Using Namespace Related Information for Opening XML Documents," U.S. Serial No. 10/185,940, filed June 27, 2002, both U.S. patent applications of which are incorporated herein by reference as if fully set out herein. For a detailed description of a mechanism for downloading software components such as XML schema files and associated solutions from a Namespace or schema library, see US Patent Application entitled Mechanism for Downloading Software Components from a Remote Source for Use by a Local Software Application, US Serial No. 10/164,260, filed June 5, 2002.

Referring still to Fig. 3, a schema validation functionality module 350 is illustrated for validating XML markup applied to a document 310 against an XML schema file 330 attached to or otherwise associated with the document 310, as described above. As described above, the schema file 330 sets out acceptable XML elements and associated attributes and defines rules for the valid annotation of the document 310 with XML markup from an associated schema file 330. For example, as shown in the schema file 330, two child elements <title> and <body> are defined under the root or parent element <intro card>. Attributes 340, 345 defining the acceptable string length of text associated with the child elements <title> and <body> are also illustrated. As described above, if a user attempts to annotate the document 310 with XML markup from a schema file 330 attached to or associated with the document in violation of the XML markup definitions contained in the schema file 330, an invalidity or error state will be presented. For example, if the user attempts to enter a title string exceeding twenty-five characters, that text entry will violate the maximum character length attribute of the <title> element of the schema file 330. In order to validate XML markup applied to a document 310, against an associated schema file 330, a schema validation module 350 is utilized. As should be understood by those skilled in the art, the schema validation module 350 is a software module including computer executable

instructions sufficient for comparing XML markup and associated text entered in to a document 310 against an associated or attached XML schema file 330 as the XML markup and associated text is entered in to the document 310.

According to embodiments of the present invention, the schema validation module 350 compares each XML markup element and associated text or data applied to the document 310 against the attached or associated schema file 330 to determine whether each element and associated text or data complies with the rules and definitions set out by the attached schema file 330. For example, if a user attempts to enter a character string exceeding twenty-five characters annotated by the <title> elements 320, the schema validation module will compare that text string against the text string attribute 340 of the attached schema file 330 and determine that the text string entered by the user exceeds the maximum allowable text string length. Accordingly, an error message or dialogue will be presented to the user to alert the user that the text string being entered by the user exceeds the maximum allowable character length according to the attached schema file 330. Likewise, if the user attempts to add an XML markup element between the <title> and the <body> elements, the schema validation module 350 will determine that the XML markup element applied by the user is not a valid element allowed between the <title> and <body> elements according to the attached schema file 330. Accordingly, the schema validation module 350 will generate an error message or dialogue to the user to alert the user of the invalid XML markup.

Context Free Document Text and Data Portions

As briefly described above, embodiments of the present invention provide methods and systems for inserting a portion of text or data into a document where the inserted portion of text or data carries with it its own specific formatting and style properties such as specified styles, fonts, list types, and the like. Fig. 4 is a block diagram illustrating interaction between a first document and a second document where a portion of text is copied or moved from the second document to the first document according to embodiments of the present invention. Fig. 5 is a block diagram

illustrating interaction between a first document and a second document where a portion of text is copied or moved from the second document to the first document according to embodiments of the present invention. It should be understood that describing the present invention in terms of copying or moving a text portion from one document to another is by way example only. Apart from copying or moving text from one document to another, the self-describing properties of the text portion allow users and software applications who desire to or who are required to parse and manipulate a file to apply rich editing to portions of the file or document without having to parse the entire document to manage properties associated with the entire document or file.

Referring to Fig. 4, a document 410 and a document 420 are described for purposes of illustrating how a portion of text may be copied from the document 420 into the document 410 according to embodiments of the present invention. The first document 410 has a title and two paragraphs. Also illustrated for the document 410 is an XML structure pane 425 showing Extensible Markup Language (XML) structure applied to the document 435. For example, a <title> element is applied to the title of the document 435, a <paragraph 1> element is applied to the first paragraph of the document 435 and a <paragraph 2> element is applied to the second paragraph of the document 435. The second document 420 includes a document 440 that has been similarly marked up with XML structure. An XML structure pane 430 is illustrated for the document 420.

According to embodiments of the present invention, a text or data portion from the second document may be copied or moved to the first document such that style and other formatting properties are carried with the text or data portion from the one document to another document. Accordingly, there is no need to resolve style or formatting properties associated with the text portion inserted into the first document against style or other formatting properties associated with the first document. For example, as shown in Fig. 4 the first document may have a header information properties element for referencing styles and other formatting properties associated with the first document. That is, the user or creator of the first document may have established a set of styles or formatting properties for the first document. According to

prior art systems, as described above, if the user desires to copy the second paragraph 455 from the second document 420 into the first document 410, style or formatting information associated with the second paragraph 455 in the second document 420 may conflict with style or formatting information contained at the document level of the first document 410.

For example, the first document 410 may have a style called "Header 1" causing a formatting of boldface and italics to be applied to the document 435. On the other hand, the creator of the second document 420 may have likewise used the style designator "Header 1" for formatting the second paragraph 455 containing the second document to include boldfacing, but not italics. If the second paragraph 455 of the second document 420 is copied into the first document 410, the style applied to the second document will conflict with the style applied to the document in the first document 410. Therefore, the user of the first document 410 will be required to manually parse the header information properties element of the XML structure applied to the first document 410 in order to make changes to the styles applied to the inserted text or data in order to prevent a conflict of the two styles.

According to embodiments of the present invention, an XML schema is provided for applying style markup as specified text and data portions of a document such as the second paragraph 455 of the second document 420 so that particular styles or formatting applications to a given text or data portion are carried with the text or data portion to the first document. By applying an XML markup to a particular text or data selection, the first document need not de-conflict style or formatting properties applied to inserted text or data portions so long as the first document has access to the style or formatting markup applied to the inserted text or data portion.

Referring to Fig. 5, a text portion 510 and a text portion 520 are illustrated for inserting into the first document 410. According to embodiments of the present invention, each text portion 510 and 520 may be a sentence, a paragraph, multiple paragraphs, fragments of sentences or paragraphs, alphanumeric data, or any other native objects. As should be appreciated, the creator of the second document 420 marks up the second document with Extensible Markup Language data according to the

context free text portion schema which defines XML elements, as described below, for specifying particular text or data portions which carries with them their own styles or other formatting properties. In effect, each of the text portions 510 and 520 become "mini" documents carrying their own style and formatting properties that may be inserted into a first document 410 and which may be treated separately from the main document 410 by an XML parsing application operating on the document 410.

It should be understood, a text or data portion such as the second paragraph 455 or either of the text portions 510, 520 illustrated in Fig. 5, may be copied and pasted or cut and pasted from the second document into the first document according to conventional cut/copy and pasting methods available from a variety of software applications. Alternatively, according to a variety of Extensible Markup Language parsing applications, the user may likewise insert a text or data portion from a second document into a first document by selecting an appropriate XML node from the XML structure pane 430 and copying that XML node into the XML structure of the first document. That is, by selecting the <paragraph 2> node 460 from the XML structure 430 of the second document 420 and by moving that node to the XML structure of the first document 410, the XML markup and associated text or data of the second paragraph 455 may be inserted into the first document along with its own style and other formatting properties as described herein.

To further illustrate the operation of embodiments of the present invention, the following XML structure is associated with a simple document having the following two paragraphs.

Para 1: The quick brown fox jumps over the lazy dog.

Para 2: The quick brown fox jumps over the lazy dog.

As can be seen in the following XML structure, a font of "Times New Roman" and a style identification of "Default Paragraph Font" is applied to the two paragraphs. The actual text of the two paragraphs is also illustrated in the following XML structure.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
-<w:wordDocument
    xmlns:w="http://schemas.microsoft.com/office/word/2003/wordml"
    xmlns:wx="http://schemas.microsoft.com/office/word/2003/auxHint"
    w:macrosPresent="no"  w:embeddedObjPresent="no"  w:ocxPresent="no"
    xml:space="preserve">

  - <w:styles>
      <w:versionOfBuiltInStylenames w:val="4" />
      <w:latentStyles w:defLockedState="off" w:latentStyleCount="156" />

    - <w:style w:type="paragraph" w:default="on" w:styleId="Normal">
        <w:name w:val="Normal" />
      - <w:rPr>
          <wx:font wx:val="Times New Roman" />
          <w:sz w:val="24" />
          <w:sz-cs w:val="24" />
          <w:lang w:val="EN-US" w:fareast="EN-US" w:bidi="AR-
            SA" />
        </w:rPr>
      </w:style>
    -       <w:style       w:type="character"       w:default="on"
        w:styleId="DefaultParagraphFont">
        <w:name w:val="Default Paragraph Font" />
        <w:semiHidden />
      </w:style>
    - <w:style w:type="list" w:default="on" w:styleId="NoList">
        <w:name w:val="No List" />
        <w:semiHidden />
      </w:style>
```

```
</w:styles>
- <w:body>
  - <w:p>
    - <w:r>
        <w:t>The quick brown fox jumps over the lazy dog.</w:t>
      </w:r>
    </w:p>
  - <w:p>
    - <w:r>
        <w:t>The quick brown fox jumps over the lazy dog.</w:t>
      </w:r>
    </w:p>
    <w:p />
    </w:body>
</w:wordDocument>
```

According to embodiments of the present invention, consider for example that the user would like to insert a third paragraph into the document as follows "The quick brown fox jumps over the lazy dog" so that the inserted paragraph provides a final document as follows.

Para 1: The quick brown fox jumps over the lazy dog.

Para 2: The quick brown fox jumps over the lazy dog.

Para 3: *The quick brown fox jumps over the lazy dog.*

The following is an example XML structure showing the first two paragraphs of the document with the normal default style applied to them. According to this example, as shown in the following XML structure, the user inserts the third italicized paragraph into the document under a style called "My Style." Because the paragraph is being inserted in an environment outside of the native Word Processing application, the logic of checking that the "My Style" style exists must be done

manually. The XML file must be parsed to find the proper location of the style declarations, and then the style declarations must be parsed to guarantee that there is no conflict. Once this has been done, the "My Style" declaration can be entered. Now that the "My Style" declaration is entered, then the XML file must be parsed again to find the proper location for the paragraph itself. Since there was no conflict with the "My Style" declaration, the paragraph being inserted can still reference the "My Style" style. Otherwise, the paragraph itself would have to be altered to reference the proper style.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<?mso-application progid="Word.Document"?>
- <w:wordDocument
  xmlns:w="http://schemas.microsoft.com/office/word/2003/wordml"
  xmlns:wx="http://schemas.microsoft.com/office/word/2003/auxHint"
  w:macrosPresent="no" w:embeddedObjPresent="no" w:ocxPresent="no"
  xml:space="preserve">
 - <w:styles>
    <w:versionOfBuiltInStylenames w:val="4" />
    <w:latentStyles w:defLockedState="off" w:latentStyleCount="156" />

   - <w:style w:type="paragraph" w:default="on" w:styleId="Normal">
      <w:name w:val="Normal" />
     - <w:rPr>
        <wx:font wx:val="Times New Roman" />
        <w:sz w:val="24" />
        <w:sz-cs w:val="24" />
        <w:lang w:val="EN-US" w:fareast="EN-US" w:bidi="AR-
          SA" />
      </w:rPr>
    </w:style>
```

<w:style w:type="character" w:default="on" w:styleId="DefaultParagraphFont">
<w:name w:val="Default Paragraph Font" />
<w:semiHidden />
</w:style>
<w:style w:type="table" w:default="on" w:styleId="TableNormal">
<w:name w:val="Normal Table" />
<wx:uiName wx:val="Table Normal" />
<w:semiHidden />
<w:rPr>
<wx:font wx:val="Times New Roman" />
</w:rPr>
<w:tblPr>
<w:tblInd w:w="0" w:type="dxa" />
<w:tblCellMar>
<w:top w:w="0" w:type="dxa" />
<w:left w:w="108" w:type="dxa" />
<w:bottom w:w="0" w:type="dxa" />
<w:right w:w="108" w:type="dxa" />
</w:tblCellMar>
</w:tblPr>
</w:style>
<w:style w:type="paragraph" w:styleId="myStyle">
<w:name w:val="myStyle" />
<w:basedOn w:val="Normal" />
<w:rsid w:val="000874C3" />
<w:pPr>
<w:pStyle w:val="myStyle" />
</w:pPr>
<w:rPr>

```
            <wx:font wx:val="Times New Roman" />
            <w:color w:val="FF0000" />
          </w:rPr>
        </w:style>
      - <w:style w:type="list" w:default="on" w:styleId="NoList">
          <w:name w:val="No List" />
          <w:semiHidden />
        </w:style>
      </w:styles>
    - <w:body>
      - <w:p>
        - <w:r>
            <w:t>The quick brown fox jumps over the lazy dog.</w:t>
          </w:r>
        </w:p>
      - <w:p>
        - <w:r>
            <w:t>The quick brown fox jumps over the lazy dog.</w:t>
          </w:r>
        </w:p>
      - <w:p>
        - <w:pPr>
            <w:pStyle w:val="myStyle" />
          </w:pPr>
        - <w:r>
            <w:t>The quick brown fox jumps over the lazy dog.</w:t>
          </w:r>
        </w:p>
        <w:p />
      </w:body>
```

</w:wordDocument>

As described above, according to embodiments of the present invention, an XML schema is provided which provides an XML element that may be used to markup and describe a particular text or data portion of a document such that style or other formatting properties applied to that text or data portion are treated separately from the remaining style or formatting properties associated with a document into which the text or data portion is inserted. So long as the document into which the text or data portion is inserted is operated by an application that may understand the XML schema associated with the text or data portion inserted into the document, the XML element wrapping the inserted text or data portion will be understood.

Below is a sample XML structure showing the addition of the third italicized paragraph "The quick brown fox jumps over the lazy dog" according to embodiments of the present invention. As shown in the following XML structure, the first two paragraphs are included in the document having style identification of "Times New Roman" and "Default Paragraph Font." However, as shown in the following XML structure, the third paragraph being inserted into the document has been wrapped in an XML element called <cfChunk>. As should be appreciated, the element <cfChunk> is illustrative of an infinite number of names that could be provided to the element according to embodiments of the present invention by the creator of this particular XML structure. As should be understood, the element might similarly be called context free portion, CFportion, context free data, CFdata, and the like. Inside the element <cfChunk> is contained the third paragraph "The quick brown fox jumps over the lazy dog" having a style identification of "My Style" and having a font identification of "italics."

According to embodiments of the present invention, when the third paragraph "The quick brown fox jumps over the lazy dog" is copied or moved from a second document into a first document, the text or data portion moved points to or references an XML schema file that provides an XML parsing application associated with the first document access to the grammatical and other definitional rules associated

with XML elements according to the schema file including the <cfChunk> element utilized for wrapping a text or data portion and for carrying with the text or data portion its own individual styles or other formatting properties. Accordingly, when the first document's XML parsing application encounters the element <cfChunk> that application may refer to the associated XML schema to learn that the style and other formatting designations contained in the structure wrapped inside the <cfChunk> element is to be treated particularly for a text or data portion being inserted into the first document and being associated with the <cfChunk> element. Accordingly, when the third paragraph is inserted into the first document according to embodiments of the present invention, as illustrated in the following sample XML structure, the style and other formatting properties associated with that text portion are honored by the application receiving the inserted text or data portion without regard to potentially conflicting style or other formatting properties associated with the document into which the text portion is inserted. The application does the necessary work to resolve any conflicts, and to properly associate the style reference for the paragraph with the appropriate style.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
- <w:wordDocument
    xmlns:w="http://schemas.microsoft.com/office/word/2003/wordml"
    xmlns:wx="http://schemas.microsoft.com/office/word/2003/auxHint"
    w:macrosPresent="no"  w:embeddedObjPresent="no"  w:ocxPresent="no"
    xml:space="preserve">
  - <w:styles>
      <w:versionOfBuiltInStylenames w:val="4" />
      <w:latentStyles w:defLockedState="off" w:latentStyleCount="156" />

    - <w:style w:type="paragraph" w:default="on" w:styleId="Normal">
        <w:name w:val="Normal" />
      - <w:rPr>
```

```
<wx:font wx:val="Times New Roman" />
<w:sz w:val="24" />
<w:sz-cs w:val="24" />
<w:lang w:val="EN-US" w:fareast="EN-US" w:bidi="AR-
SA" />
</w:rPr>
</w:style>
<w:style w:type="character" w:default="on"
w:styleId="DefaultParagraphFont">
<w:name w:val="Default Paragraph Font" />
<w:semiHidden />
</w:style>
<w:style w:type="list" w:default="on" w:styleId="NoList">
<w:name w:val="No List" />
<w:semiHidden />
</w:style>
<w:style w:type="table" w:default="on" w:styleId="TableNormal">
<w:name w:val="Normal Table" />
<wx:uiName wx:val="Table Normal" />
<w:semiHidden />
<w:rPr>
<wx:font wx:val="Times New Roman" />
</w:rPr>
<w:tblPr>
<w:tblInd w:w="0" w:type="dxa" />
<w:tblCellMar>
<w:top w:w="0" w:type="dxa" />
<w:left w:w="108" w:type="dxa" />
<w:bottom w:w="0" w:type="dxa" />
<w:right w:w="108" w:type="dxa" />
```

```
            </w:tblCellMar>
          </w:tblPr>
        </w:style>
      </w:styles>
- <w:body>
  - <w:p>
    - <w:r>
        <w:t>The quick brown fox jumps over the lazy dog.</w:t>
      </w:r>
    </w:p>
  - <w:p>
    - <w:r>
        <w:t>The quick brown fox jumps over the lazy dog.</w:t>
      </w:r>

    </w:p>
  - <w:cfChunk>
    - <w:styles>
      - <w:style w:type="paragraph" w:styleId="myStyle">
          <w:name w:val="myStyle" />
          <w:basedOn w:val="Normal" />
          <w:rsid w:val="000874C3" />
        - <w:pPr>
            <w:pStyle w:val="myStyle" />
          </w:pPr>
        - <w:rPr>
            <wx:font wx:val="Italics" />
            <w:color w:val="FF0000" />
          </w:rPr>
        </w:style>
```

```
        </w:styles>
      - <w:p>
        - <w:pPr>
            <w:pStyle w:val="myStyle" />
          </w:pPr>
        - <w:r>
            <w:t>The quick brown fox jumps over the lazy
              dog.</w:t>
          </w:r>
        </w:p>
      </w:cfChunk>
      <w:p />
    </w:body>
  </w:wordDocument>
```

As described herein, self-describing portions of text or data are provided for inserting into a document whereby an Extensible Markup Language schema is provided for describing XML markup that may be applied to a text or data portion to allow particular style or other formatting properties applied to the text or data portion to travel with the text or data portion to a different document or to a different location in a same document. It will be apparent to those skilled in the art that various modifications or variations may be made in the present invention without departing from the scope or spirit of the invention. Other embodiments of the invention will be apparent to those skilled in the art from consideration of the specification and practice of the invention disclosed herein.